Journal of Computational and Applied Mathematics 233 (2010) 3139–3147

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Comparison of refinement criteria for structured adaptive mesh refinement

Shengtai Li

Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, United States

ARTICLE INFO

Article history:
Received 10 April 2009
Received in revised form 17 August 2009

Keywords:
Adaptive mesh refinement (AMR)
Refinement criteria
Verification and validation

ABSTRACT

We have investigated and analyzed the grid convergence issues for an adaptive mesh refinement (AMR) code. We have found that the numerical results for the AMR grid may have a larger error than those for the unrefined uniform grid. After a detailed analysis, we have found that the numerical solution at the coarse–fine interface between different levels of the grid converges only in the first-order accuracy. Therefore, the error near the coarse–fine interface can quickly dominate the error in the other regions if the coarse–fine interface is active and not covered by the fine grid. We propose, implement, and compare several refinement criteria. Some of them can catch the large-error region near the coarse–fine interface and refine them with the fine grid.

Published by Elsevier B.V.

1. Introduction

Convergence analysis of numerical solutions to partial differential equations (PDEs) is typically performed on a uniform fixed mesh [1]. Such regular meshes greatly simplify both the method and the implementation of the asymptotic convergence analysis. A previous report [2] extends the methodology to Lagrangian staggered grid hydrodynamics algorithms that involve unstructured dynamically evolving meshes. Adaptive mesh refinement (AMR), which locally adds the cells to where they are needed, is straightforward in theory. In this article, we focus on the structured AMR [3,4] that has been used for Eulerian compressible hydrodynamics simulations. The structured AMR consists of logically rectangular grids of different refinement levels. AMR has been popular in the past 20 years for its capabilities to handle problems with multiple length and time scales. However, the verification analysis for AMR is still in its infancy. The reason is partially due to the complexity of the AMR data structure and its detachment from the numerical algorithm.

In the past several years, we have developed a structured AMR package, AMR-MHD, for simulations of both hydro and magneto-hydrodynamics (MHD) flows. AMR-MHD [5] uses a patch-based framework and allows local smaller time steps for a locally finer grid patch. In AMR-MHD, we use the cell-average quantities as our variables, and the flux is calculated at the cell-faces using the Riemann solver with the left and right states predicted by Colella's second-order method [6]. We have performed code verification by using the AMR-MHD code to solve the model problems with exact solutions. All of the errors depicted in the plots are calculated at the cell center.

The outline of the article is as follows. In Section 2, we present results for AMR and compare them with those for a uniform grid without AMR. In Section 3, we analyze the results and provide some insight into the large error that occurs at the fine–coarse interface. In Section 4, we propose several new refinement criteria and test them using the model problem.

E-mail address: sli@lanl.gov.

0377-0427/$ – see front matter. Published by Elsevier B.V.
doi:10.1016/j.cam.2009.08.104

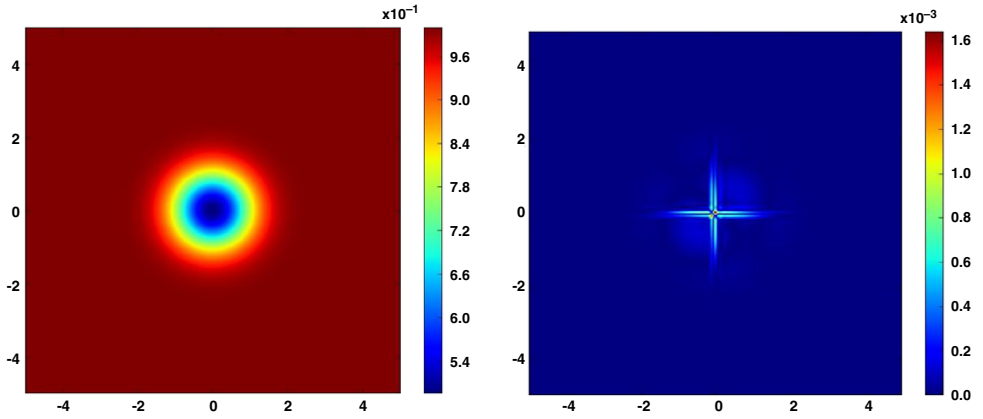

Fig. 1. Solution and numerical error of AMR-MHD for 640×640 uniform grid results at $t = 10$. The left one is the color plot for the density. The right one is the error distribution. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2. Model problem and verification results for AMR

We solve the compressible Euler equations of gas dynamics in 2D. The problem is subject to the following initial conditions:

$$\rho_{t=0} = \left(1 - \frac{(\gamma-1)\epsilon^2}{8\gamma\pi^2} e^{1-r^2}\right)^{1/(\gamma-1)}$$
$$(u, v, p)_{t=0} = (1, 1, 1),$$

where $r^2 = x^2 + y^2$ and $\epsilon = 5$. The density is chosen so that only a small region needs refinement, if AMR is used. The computational domain is taken as $[-5, 5] \times [-5, 5]$, extended periodically in both the directions. The exact solutions for Euler equations with the above initial and boundary conditions are just a linear advection of the density profile with the constant velocity $(1, 1)$. The solution is very smooth; no shocks are developed. We first solve it with four uniform grids: 80×80, 160×160, 320×320, and 640×640. Then, we solve it using the AMR grid with different refinement levels and different base grids. We will denote AMR with n level refinement (total $n + 1$ level if the base grid is included) as "AMR(n)". Since we have exact solutions at any time, we can exactly calculate the numerical error.

We have verified that our numerical results converge in second order for the uniform grid (see [7] for more details). In this article, we investigate the accuracy of the AMR grid. We will run the simulation for one period to $t = 10$.

Fig. 1 shows the solution and the numerical error calculated with 640×640 grid. Note that the largest error occurs at the middle of the domain.

For AMR calculation, we use the refinement criteria suggested by the FLASH code [8]. Flash is another AMR code, developed by the Flash center of the University of Chicago. Flash uses the modified second derivative error criterion,

$$E_i = \frac{|u_{i+1} - 2u_i + u_{i-1}|}{|u_{i+1} - u_i| + |u_i - u_{i-1}| + \epsilon[|u_{i+1}| + 2|u_i| + |u_{i-1}|]}, \tag{1}$$

for a 1D uniform mesh, where the constant ϵ is given a value of 0.01. This criterion can be extended to multi-dimensions. If the maximum error norm (E_i) on a block is larger than an adjustable constant, CTORE, the block is marked for refinement. If the error norm is less than a second constant CTODE, the block is marked for de-refinement. Currently, Flash sets CTORE $= 0.8$ and CTODE $= 0.2$. In AMR-MHD, we have only one tolerance, CTORE, to flag the cell that needs to be refined. The de-refinement is automatically done when the cell is not flagged.

Fig. 2 shows the AMR results calculated with different base grids and refinement levels. It is clear that the large error occurs near the fine–coarse interface. Note that our refinement tolerances are already far smaller than the suggested value (CTORE $= 0.8$) by FLASH code. If we use CTORE $= 0.8$, there will be no refinement at all for this specific problem.

By comparing the error distribution in Figs. 1 and 2, we find that the large error near the coarse–fine interface is purely due to the refinement.

3. Modified equation analysis and source of errors

A detailed analysis of the numerical errors in report [7] shows that we achieve only first-order convergence for the AMR results. Since the dominant errors occur at the coarse–fine interface regions in the AMR results, the following derivation suggests that we achieve only first-order convergence at the coarse–fine interface theoretically.

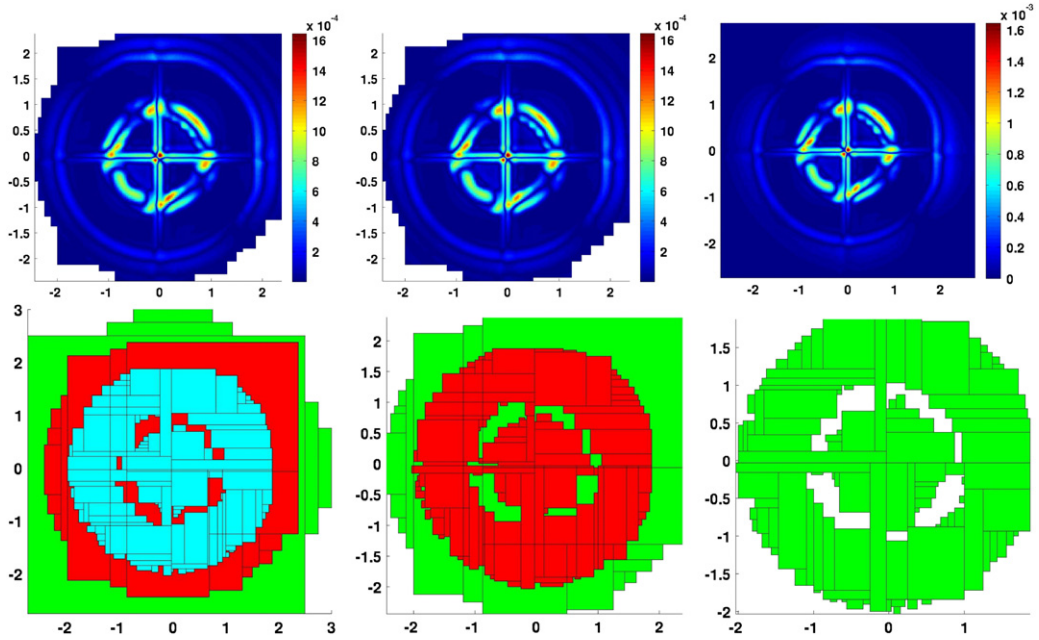

Fig. 2. Numerical error (top) and refinement (bottom) of AMR-MHD with Flash refinement criteria (1). From left to right, the plots are for 80 × 80 grid with AMR(3), 160 × 160 grid with AMR(2), and 320 × 320 grid with AMR(1). CTORE = 0.02 is used. Color map for the refinement: second level (green), third level (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

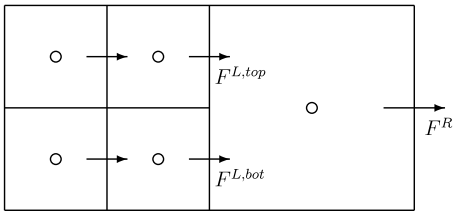

Fig. 3. Local truncation errors at the coarse–fine interface.

Fig. 3 illustrates the local truncation errors at the coarse–fine interface. Assume that the flux evaluation is exact. Then the divergence for the numerical flux at the coarse–fine interface is

$$\begin{aligned} D^*F &= \frac{1}{\Delta x}\left(F^R - \frac{1}{2}\left(F^{L,\text{top}} + F^{L,\text{bot}}\right)\right) \\ &= \frac{1}{\Delta x}\left(F\left(\left(\text{i}+\frac{1}{2}\right)\Delta x, \bar{y}\right) - \left(F\left(\left(\text{i}-\frac{1}{2}\right)\Delta x, \bar{y}\right) + C(\Delta x)^2\right)\right) \\ &= \frac{\partial F}{\partial x} + O(\Delta x) \quad (\textbf{not } O(\Delta x^2)). \end{aligned}$$

Using the modified equation analysis, we obtain for nonlinear time-dependent problems:

$$\begin{aligned} \frac{\partial U^{\text{mod}}}{\partial t} + \nabla \cdot F^{\text{mod}} = \tau &= O(\Delta x) \quad \text{at C/F boundary} \\ &= O(\Delta x^2) \quad \text{otherwise.} \end{aligned}$$

Although the error is first order only at the coarse cells at the coarse–fine interface, the truncation error can propagate to the nearby cells, including the fine cells.

Note that the truncation error $O(h^p)$ is not only determined by the order of accuracy (p), but also the coefficients in front of h^p. It is possible that $O(h^p)$ is smaller for cells with $p = 1$ than for cells with $p = 2$. That is why AMR is still popular in scientific computing. We remark that AMR performs very well for the problems with discontinuity, where the accuracy near the discontinuity is not more than the first order. However if some smooth feature (e.g., vortex) requires refinement, then the large error near the refinement boundary will be triggered. The remaining question is how to catch those regions and refine them. They usually locate at the fine–coarse boundaries and are difficult to catch by an engineering approach, which uses the magnitude of some sets of solution derivatives.

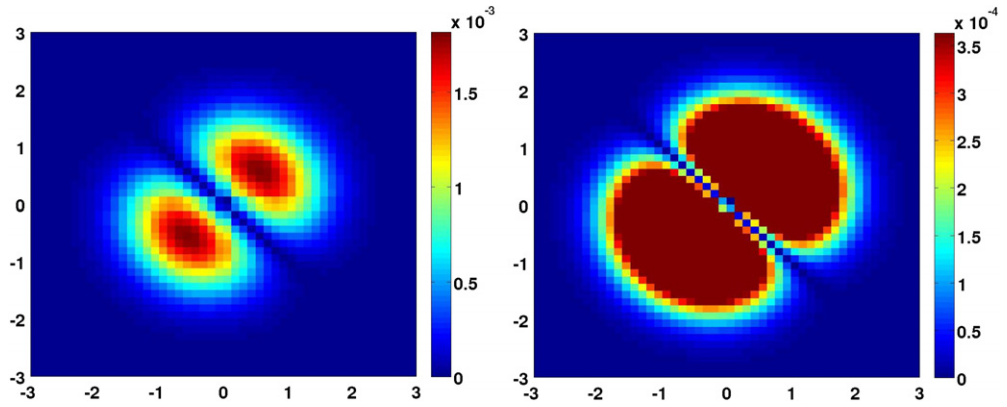

Fig. 4. Initial local truncation error estimated by (2). The right figure shows the refinement region (saturated with red color) when the error threshold is 9.67e−4, which is about 20% of the maximum error. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4. Other refinement criteria

In this subsection, we investigate if any refinement criteria can catch those large-error regions depicted in Fig. 2. We first investigate the truncation error estimator based on the Richardson extrapolation.

4.1. Local truncation error estimator by the Richardson extrapolation

Berger [3] proposed an approximate error estimator for the local truncation error using the Richardson extrapolation. Her approach relies on the regularity of the structured grids. It is easy to implement in a patch-based AMR. We illustrate it with a 1D example. If the solution is smooth enough, the local truncation error has a leading term, denoted by τ, that satisfies

$$u(x_i, t^{n+1}) - Q_h u(x_i, t^n) = \tau(x, t, h, k) + h.o.t.,$$

where the numerical method Q_h has an order of accuracy q in both time and space; h and k are spatial step and time step. If two time steps are considered, then the leading error is doubled

$$u(x_i, t^{n+2}) - Q_h Q_h(x_i, t^n) = 2\tau(x, t) + h.o.t.$$

Let Q_{2h} denote the same difference method as Q, but based on a mesh width of $2h$ and time step $2k$. Then:

$$u(x_{i+1/2}, t^{n+2}) - Q_{2h} u(x_{i+1/2,t^n}) = 2^{p+1}\tau + h.o.t.,$$

where $x_{i+1/2} = \frac{1}{2}(x_i + x_{i+1})$ is the cell-centered mesh with $2h$ spacing and p is the order of the numerical scheme. The comparison

$$\frac{(Q_h^2 u(x_i, t^n) + Q_h^2 u(x_{i+1}, t^n))/2 - Q_{2h} u(x_{i+1/2}, t^n)}{2^{p+1} - 2} = \tau + h.o.t., \tag{2}$$

provides an estimate of the local truncation error at time t. Note that the exact form of the truncation error is not necessary, only its order is required.

We have implemented this approach in AMR-MHD. The initial local truncation error for Berger's approach (2) is shown in Fig. 4. Even with a small threshold, we still find that some region is not refined if refinement is done cell by cell. However, patch-based AMR will refine those regions between the large errors because it includes nearby unflagged cells during the patch generation.

Berger's approach is better used for an unsplit solver, which has a uniform order of accuracy in each step. We test the error estimator (2) with Colella's multi-dimensional unsplit solver. The error threshold for refinement is set to CTORE $= 10^{-4}$. We use only one-level refinement and the base grid is 160×160. Fig. 5 shows the error distribution and the mesh refinement after one period (at $t = 10$). For comparison, we also show the plots for Flash's error monitor (1) with CTORE $= 0.05$. It is clear that the error estimator (2) is far better than (1).

The error estimator (2) is well defined to catch the large truncation error region. However, its implementation is limited by several factors. As suggested in [3], the estimator (2) relies highly on the regularity of the structured grids. For an unstructured grid the implementation may require a huge effort. It requires not only the temporary generation of a coarse mesh, but also the use of the integrator (a time-stepper) during the mesh generation. Since many AMR packages, such as FLASH and RAGE, use the locked time step to advance all the cells at the same time with the same time step, it is difficult

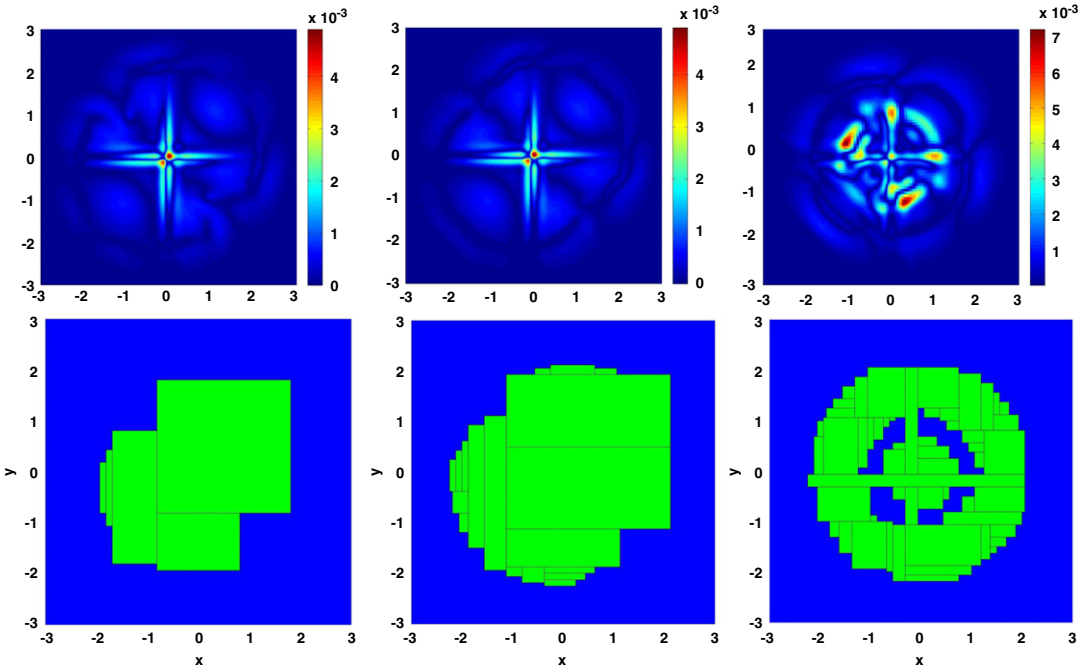

Fig. 5. Comparison of AMR refinement and error distribution for three different monitors after one period at $t = 10$. The left plot is for the local truncation error estimator (2) with refinement threshold 10^{-4}. The right plot is for the error estimator (1) with CTORE $= 0.05$. The middle plot is for the error monitor (8) with refinement threshold 0.01. 160×160 base grid with one-level refinement is used.

to solve the equations only on a coarse mesh and compare their results with the solutions on the usual mesh, because the mesh must be generated initially for it to be solved. Berger's original AMR allows the local time step and the integration is done recursively with the coarse-level mesh first and then with the fine-level mesh. Therefore, it is easy to implement and generate a new fine-level mesh after a coarse-level integration. Berger claims that the error estimator is inexpensive, because it is only applied on coarse-level grids. However, for many AMR packages, especially parallel AMR packages, the coarse-level grids cannot be singled out during the time integration.

4.2. Operator recovery error source detector

Gradient recovery error estimators (GREE) [9] are widely used in the finite-element community. Rigorously, the GREE are valid only for elliptic problems. Lapenta [10] proposed a new refinement criteria based on a generalization of the GREE and valid for any differential operator. His method is called *operator recovery error source detector* (ORESD).

The ORESD for the gas dynamics equations should consider three equations, separately. Lapenta [10] proposed a simple method to combine the error source detectors for various equations and use only the internal energy equation. In an Eulerian frame, the internal energy equation is

$$\frac{De}{Dt} + \frac{p}{\rho}\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla e = 0, \tag{3}$$

where $\mathbf{u}$ is the velocity vector, e is the specific internal energy, and the derivative operator, D, represents the total derivatives. The ORESD for gas dynamics can be constructed based on the operator $\frac{p}{\rho}\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla e$.

For the sake of completeness, we outline the implementation in [10] for Eq. (3) here and refer readers to Ref. [10], for more details. In Ref. [10], the cell-centered values are constructed based on node-centered values. Since we know the cell-centered values in our solver, we will reconstruct the node-centered values. This shows the difference from the implementation of Ref. [10]. For a structured grid, such as a patch in AMR-MHD, the reconstruction is easy to implement. As suggested in Ref. [10], we first compute the node-centered values by linear averaging:

$$\mathbf{u}_n = \frac{1}{N_n}\sum_{c(n)} \mathbf{u}_c, \qquad \rho_n = \frac{1}{N_n}\sum_{c(n)} \rho_c, \qquad p_n = \frac{1}{N_n}\sum_{c(n)} p_c, \qquad e_n = \frac{1}{N_n}\sum_{c(n)} e_c,$$

where the summation is over the N_n cells around node n. Next, we compute the cell-centered operator:

$$\mathcal{O}_c = \frac{p_c}{\rho_c}(\nabla \cdot \mathbf{u})_c + \mathbf{u}_c \cdot (\nabla e)_c,$$

and the node-centered operator:

$$\mathcal{O}_n = \frac{p_n}{\rho_n}(\nabla \cdot \mathbf{u})_n + \mathbf{u}_n \cdot (\nabla e)_n.$$

From the node-centered operator, we reconstruct the cell-centered operator via the linear interpolation within the cell:

$$\tilde{\mathcal{O}} = \sum_{n(c)} \mathcal{O}_n S(\mathbf{x}_n - \mathbf{x}),$$

where $S(\mathbf{x}_n - \mathbf{x})$ are the base functions for linear interpolations. For a 2D structured rectangular grid, a base function at any node for a cell is a compact-support hat function, defined by:

$$S_n(x, y) = c_0 + c_1(x - x_n) + c_2(y - y_n) + c_3(x - x_n)(y - y_n),$$

where (x_n, y_n) is the coordinate of node $\mathbf{x}_n$, and c_i are the coefficients to be determined. Since the values of $\mathcal{O}_n$ at four nodes of a cell are already known, c_i can be obtained easily. Finally, the operator recovery error is defined as:

$$\mathrm{err}^p = \frac{1}{V_c}\int_{V_c} (|\tilde{\mathcal{O}} - \mathcal{O}_c|)^p \mathrm{d}V. \tag{4}$$

The trick is to calculate the divergence and the gradient for the cell-centered and the node-centered values. Lapenta [10] suggests using node-centered values to calculate the divergence and the gradient at the cell center, and using cell-centered values to calculate the divergence and the gradient at the node center. Again, for a 2D structure grid, we have:

$$\begin{aligned}
(\nabla \cdot u)_{n,(i,j)} &= 0.5(\delta y(u_{c,x,(i+1,j+1)} + u_{c,x,(i+1,j)} - u_{c,x,(i,j)} - u_{c,x,(i,j+1)}) \\
&\quad + \delta x(u_{c,y,(i+1,j+1)} + u_{c,y,(i,j+1)} - u_{c,y,(i,j)} - u_{c,y,(i+1,j)}))/(\delta x \delta y), \\
(\nabla e)_{n,x,(i,j)} &= 0.5(e_{c,(i+1,j+1)} - e_{c,(i,j+1)} + e_{c,(i+1,j)} - e_{c,(i,j)})/\delta x, \\
(\nabla e)_{n,y,(i,j)} &= 0.5(e_{c,(i+1,j+1)} - e_{c,(i+1,j)} + e_{c,(i,j+1)} - e_{c,(i,j)})/\delta y,
\end{aligned}$$

where δx and δy are the local spacings. Similarly, we can construct the values at the cell center.

The method for evaluating the error in (4) is also important. We have tried three approaches:

1. L_2 norm, where $p = 2$:

$$\begin{aligned}
\mathrm{err}^2_{i,j} &= \frac{1}{9}(\mathcal{O}^2_{n,(i,j)} + \mathcal{O}^2_{n,(i,j-1)} + \mathcal{O}^2_{n,(i-1,j)} + \mathcal{O}^2_{n,(i-1,j-1)}) + \frac{1}{18}(\mathcal{O}_{n,(i-1,j)}\mathcal{O}_{n,(i,j-1)} + \mathcal{O}_{n,(i-1,j-1)}\mathcal{O}_{n,(i,j)}) \\
&\quad + \frac{1}{9}(\mathcal{O}_{n,(i,j)} + \mathcal{O}_{n,(i-1,j)})(\mathcal{O}_{n,(i,j-1)} + \mathcal{O}_{n,(i-1,j-1)}) + \mathcal{O}^2_{c,(i,j)} - \frac{1}{2}\mathcal{O}_{c,(i,j)}(\mathcal{O}_{n,(i,j)} + \mathcal{O}_{n,(i-1,j)} \\
&\quad + \mathcal{O}_{n,(i,j-1)} + \mathcal{O}_{n,(i-1,j-1)}),
\end{aligned} \tag{5}$$

   where (i, j) are the logical coordinates.
2. L_1 norm, where $p = 1$; the error has no analytic expression and must be numerically calculated. To simplify the calculation, we adopt two approximate approaches:
   (a) Original implementation from [10]:

$$\mathrm{err}_{i,j} = |0.25(\mathcal{O}_{n,(i,j)} + \mathcal{O}_{n,(i-1,j)} + \mathcal{O}_{n,(i,j-1)} + \mathcal{O}_{n,(i-1,j-1)}) - \mathcal{O}_{c,(i,j)}| \tag{6}$$

   which is valid only if all the node values are greater or smaller than $\mathcal{O}_c$. This monitor does not work well for smooth problems. We propose a modified version that uses a larger stencil:

$$\begin{aligned}
\mathrm{err}_{i,j} = \Bigg| &\frac{1}{4}(\mathcal{O}_{n,(i,j)} + \mathcal{O}_{n,(i-1,j)} + \mathcal{O}_{n,(i,j-1)} + \mathcal{O}_{n,(i-1,j-1)}) - \frac{1}{4}(\mathcal{O}_{n,(i+1,j)} - \mathcal{O}_{n,(i-1,j)} + \mathcal{O}_{n,(i,j+1)} - \mathcal{O}_{n,(i,j-1)}) \\
&+ \frac{1}{4}(\mathcal{O}_{n,(i,j)} - \mathcal{O}_{n,(i-2,j)} - \mathcal{O}_{n,(i-1,j+1)} + \mathcal{O}_{n,(i-1,j-1)}) \\
&+ \frac{1}{4}(\mathcal{O}_{n,(i,j)} - \mathcal{O}_{n,(i,j-2)} - \mathcal{O}_{n,(i+1,j-1)} + \mathcal{O}_{n,(i-1,j-1)}) \\
&+ \frac{1}{4}(\mathcal{O}_{n,(i,j-1)} - \mathcal{O}_{n,(i-2,j-1)} + \mathcal{O}_{n,(i-1,j)} - \mathcal{O}_{n,(i-1,j-2)}) - \mathcal{O}_{c,(i,j)}\Bigg|.
\end{aligned} \tag{7}$$

   (b) Modified L_1 norm:

$$\mathrm{err}_{i,j} = 0.25(|\mathcal{O}_{n,(i,j)} - \mathcal{O}_{c,(i,j)}| + |\mathcal{O}_{n,(i-1,j)} - \mathcal{O}_{c,(i,j)}| + |\mathcal{O}_{n,(i,j-1)} - \mathcal{O}_{c,(i,j)}| + |\mathcal{O}_{n,(i-1,j-1)} - \mathcal{O}_{c,(i,j)}|). \tag{8}$$

3. L_∞ norm:

$$\mathrm{err}_{i,j} = \max(|\mathcal{O}_{n,(i,j)} - \mathcal{O}_{c,(i,j)}|, |\mathcal{O}_{n,(i-1,j)} - \mathcal{O}_{c,(i,j)}|, |\mathcal{O}_{n,(i,j-1)} - \mathcal{O}_{c,(i,j)}|, |\mathcal{O}_{n,(i-1,j-1)} - \mathcal{O}_{c,(i,j)}|). \tag{9}$$

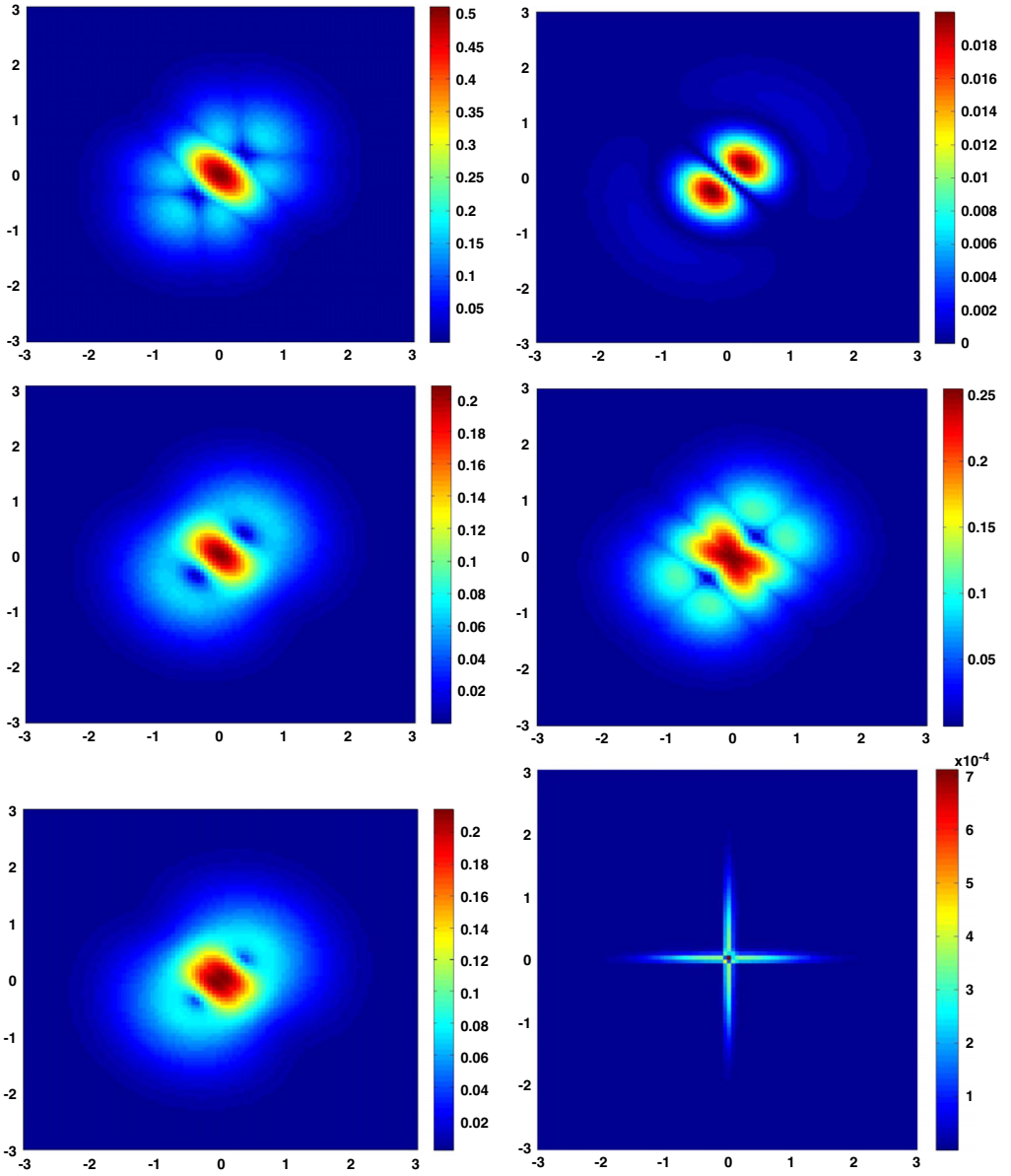

Fig. 6. Monitor values generated by the estimators (5)–(9) at $t = 0$. 160×160 uniform grid is used. The top two are for monitors (9) (left) and (7) (right). The middle two are for monitors (5) (left) and (8) (right). The bottom two are for the combination (7) and (5) (left), and the exact error distribution after one time step (right). The color bar is different for different monitors.

The error estimator (6) produces very small values ($O(10^{-14})$) for the linear wave problems and is subject to numerical noise. The error estimator (7) needs more ghost cells than others. Fig. 6 shows the error estimation for all of the monitor functions except (6). It shows that the refinement regions are similar for monitors (8), (5) and (9). All three of them have two holes along the diagonal direction and near the central region, and they should be refined. Monitor (5) has the smoothest refinement region, and monitor (8) has the largest refinement region. For patch-based AMR, such as AMR-MHD, the two holes are always refined in our simulations because the number of unflagged cells is below the threshold. However, for cell-based AMR, such as RAGE, the two holes may not be refined due to the small values of the error estimator in those cells.

It seems that the refinement region for (7) is quite different from others. It partially complements the large-error regions for other estimators. A combination of (7) with others may be better for the overall refinement (see the last column of Fig. 6, which is a result of the combination of (7) and (5). For reference, we also include the actual global error estimation after step one. It seems that the large-error regions for the numerical error estimators and the exact global error estimator are quite different. This also indicates that the global error estimator may not be a good candidate for the refinement criteria.

By comparing the plot in Fig. 6 with that of Fig. 4, we find that the large-error region for (7) is similar to that of the monitor (2), but the area with a large error in the monitor (7) is smaller and the magnitude of the maximum error is much

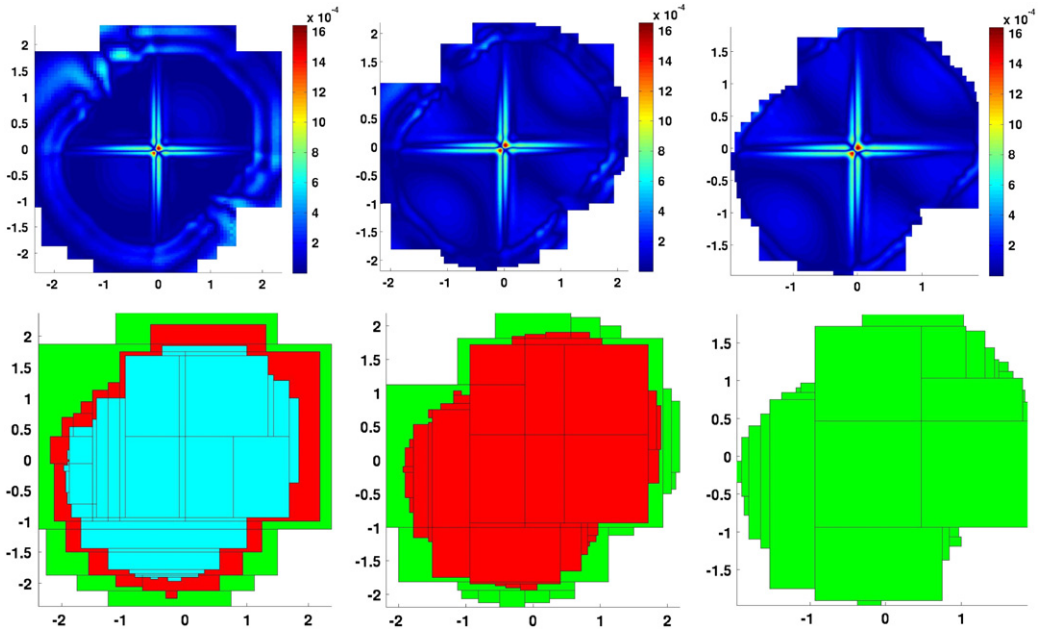

Fig. 7. Numerical errors and mesh refinement of AMR-MHD for different base grid and refinement levels. The left two are for 80 × 80 base grid with three-level refinement. The middle two are for 160 × 160 base grid with two-level refinement. The right two are for 320 × 320 base grid with one-level refinement. The top column represents the error distribution and the bottom column shows the mesh refinement region. The output is after one period at $t = 10$.

larger than that of the monitor (2). In our experience, monitor (7) alone does not work very well, even with a very small error threshold, such as the one used for (2).

Monitors (5) and (8) work similarly well on the linear wave problem. They are also insensitive to the error threshold. Since monitor (8) is much easier to evaluate and extends to 3D problems, we will test monitor (8) with error threshold 0.01. We first compare the refinement region and error distribution for monitor (8) with Berger's error estimator (2). The comparison with other monitors is shown in Fig. 5.

We have tested the monitor function (8) with more refinement levels and with different base grids. The dimensional split solver is used in all of the simulations. Fig. 7 shows the mesh refinement and the error distribution for different base grids at various refinement levels. All of them achieve the same finest resolution of 640 × 640. Therefore, we achieve the same maximum error in all the AMR simulations. One interesting aspect to be noted from the plots is that the finest level grids are very similar, despite the use of different base grids with different refinement levels.

5. Conclusion

Refinement criteria play important roles in achieving convergence results for AMR simulations. We have found that both the local truncation error estimator based on the Richardson extrapolation and the operator recovery error source detector (ORESD) work well for our chosen problem. Our experience shows that the optimal refinement criterion is still problem-dependent. We will investigate more refinement criteria with more challenging problems in the future.

Acknowledgements

This work is part of a Los Alamos National Laboratory report, *LA-UR-06-4595* [11], and was carried out under the auspices of the National Nuclear Security Administration of the US Department of Energy at the Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. The authors gratefully acknowledge the support of the Advanced Simulation & Computing program at the Los Alamos National Laboratory for funding this work as a part of the Verification and Validation Program Element under the Project Manager Jerry Brock and the Program Manager Scott Doebling.

References

[1] J.R. Kamm, W.J. Rider, J.S. Brock, Combined space and time convergence analysis of a compressible flow algorithm, AIAA-2003-4241, 2003.
[2] J.R. Kamm, J.S. Brock, C.L. Rousculp, W.J. Rider, Verification of an ASCI SHAVANO project hydrodynamics algorithm, Technical Report LA-UR-03-6999, Los Alamos National Lab, 2003.
[3] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[4] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[5] S. Li, H. Li, A modern code for solving magneto-hydrodynamic or hydrodynamic equations, Technical Report LA-UR-03-8925, Los Alamos National Lab, 2003.

[6] P. Colella, Multi-dimensional upwind methods for hyperbolic conservation laws, J. Comput. Phys. 87 (1990) 171–200.
[7] S. Li, W.J. Rider, M.J. Shashkov, Two-dimensional convergence study for problems with exact solution: Uniform and adaptive grids, Technical Report LA-UR-05-7985, Los Alamos National Lab, 2005.
[8] B. Fryxell, et al., FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes, Astrophys. J. Suppl. Ser. 131 (2000) 273–334.
[9] M. Ainsworth, J.T. Oden, A Posteriori Error Estimation in Finite Element Analysis, Wiley, New York, 2000.
[10] G. Lapenta, A recipe to detect the error in discretization schemes, Internat. J. Numer. Methods Engrg. 59 (15) (2004) 2065–2087.
[11] S. Li, W.J. Rider, Code verification and assessment for adaptive mesh refinement simulations, Technical Report LA-UR-06-4595, Los Alamos National Laboratory, 2006.